# The Evolution of Microprocessor Parallelism

More cores ➔ More Threads ➔ Wider vectors

## Intel® Xeon® Processor

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Cores</th>
<th>Threads</th>
<th>SIMD Width</th>
<th>Scalar ISA</th>
<th>SIMD ISA</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit</td>
<td>2019</td>
<td>1</td>
<td>2</td>
<td>128</td>
<td>Intel® SSE3</td>
<td>Intel® AVX</td>
</tr>
<tr>
<td>5100 series</td>
<td>2019</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>Intel® SSE3-4.1</td>
<td>Intel® AVX2</td>
</tr>
<tr>
<td>5500 series</td>
<td>2020</td>
<td>4</td>
<td>12</td>
<td>128</td>
<td>Intel® SSE4-4.1</td>
<td>Intel® AVX-512</td>
</tr>
<tr>
<td>5600 series</td>
<td>2020</td>
<td>6</td>
<td>12</td>
<td>128</td>
<td>Intel® SSE4.2</td>
<td>Intel® AVX</td>
</tr>
<tr>
<td>E5-2600 v2 series</td>
<td>2021</td>
<td>12</td>
<td>24</td>
<td>256</td>
<td>Intel® AVX</td>
<td>Intel® AVX</td>
</tr>
<tr>
<td>E5-2600 v3 series</td>
<td>2022</td>
<td>18-22</td>
<td>36-44</td>
<td>256</td>
<td>Intel® AVX</td>
<td>Intel® AVX</td>
</tr>
<tr>
<td>E5-2600 v4 series</td>
<td>2023</td>
<td>28</td>
<td>44-56</td>
<td>512</td>
<td>Intel® AVX</td>
<td>Intel® AVX</td>
</tr>
</tbody>
</table>

*1. Product specification for launched and shipped products available on arl.intel.com.*
INTEL® OPTIMIZED SOFTWARE AND LIBRARIES

Optimization techniques for getting performance

**Scaling**
- Improve load balancing
- Reduce synchronization events, all-to-all comms

**Utilize all the cores**
- OpenMP, MPI
- Reduce synchronization events, serial code
- Improve load balancing

**Vectorize/SIMD**
- Unit strided access per SIMD lane
- High vector efficiency
- Data alignment

**Efficient memory/cache use**
- Blocking
- Data reuse
- Prefetching
- Memory allocation
SPEED UP DEVELOPMENT
using open AI software

TOOLKITS

App developers

LIBRARIES

Data scientists

Open source platform for building E2E Analytics & AI applications on Apache Spark* with distributed TensorFlow*, Keras*, BigDL

Kernels

Library developers

Intel® Distribution for Python*
Intel distribution optimized for machine learning

Intel® Data Analytics Acceleration Library (Intel® DAAL)
High performance machine learning & data analytics library

Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)
Open source DNN functions for CPU / integrated graphics

Intel-optimized Frameworks

Deep learning inference deployment on CPU/GPU/FPGA/VPU for Caffe*, TensorFlow*, MXNet*, ONNX*, Kaldi*

Open source, scalable, and extensible distributed deep learning platform built on Kubernetes (BETA)

Intel® Distribution for Python*

Intel® Data Analytics Acceleration Library (Intel® DAAL)

Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)

Open source compiler for deep learning model computations optimized for multiple devices (CPU, GPU, NNP) from multiple frameworks (TF, MXNet, ONNX)

Visit: www.intel.ai/technology

Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
**PRODUCTIVITY WITH PERFORMANCE VIA INTEL® PYTHON**

*Other names and brands may be claimed as the property of others.

---

**Intel® Distribution for Python**

- Easy, out-of-the-box access to high performance Python
  - Prebuilt accelerated solutions for data analytics, numerical computing, etc.
  - Drop in replacement for your existing Python. No code changes required.

Learn More: software.intel.com/distribution-for-python
Installing Intel® Distribution for Python* 2019

Standalone Installer
Download full installer from https://software.intel.com/en-us/intel-distribution-for-python

Anaconda.org
Anaconda.org/intel channel
> conda config --add channels intel
> conda install intelpython3_full
> conda install intelpython3_core

PyPI
> pip install intel-numpy
> pip install intel-scipy
> pip install mkl_fft
> pip install mkl_random

Docker Hub
docker pull intelpython/intelpython3_full

YUM/APT
Access for yum/apt:

2.7 & 3.6
(3.7 coming soon)

Linux*
Windows*
OS X*
**FASTER PYTHON* WITH INTEL® DISTRIBUTION FOR PYTHON 2019**

**High Performance Python Distribution**

- Accelerated NumPy, SciPy, scikit-learn well suited for scientific computing, machine learning & data analytics
- Drop-in replacement for existing Python. No code changes required
- Highly optimized for latest Intel processors
- Take advantage of Priority Support – connect direct to Intel engineers for technical questions

**What’s New in 2019 version**

- Faster Machine learning with Scikit-learn: Support Vector Machine (SVM) and K-means prediction, accelerated with Intel® Data Analytics Acceleration Library
- Integrated into Intel® Parallel Studio XE 2019 installer. Also available as easy command line standalone install.
- Includes XGBoost package (Linux* only)

---

**Linear Algebra functions in Intel Distribution for Python are faster than equivalent stock Python functions**

Software & workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark & MobileMark, are measured using specific computer systems, components, software, operations & functions. Any change to any of those factors may cause the results to vary. You should consult other information & performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to [http://www.intel.com/performance](http://www.intel.com/performance).

---

*Other names and brands may be claimed as the property of others.

---

Software & workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark & MobileMark, are measured using specific computer systems, components, software, operations & functions. Any change to any of those factors may cause the results to vary. You should consult other information & performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to [http://www.intel.com/performance](http://www.intel.com/performance).
INTEL® DATA ANALYTICS ACCELERATION LIBRARY (INTEL® DAAL)

Building blocks for all data analytics stages, including data preparation, data mining & machine learning

- Pre-processing
- Transformation
- Analysis
- Modeling
- Validation
- Decision Making

Common Python*, Java*, C++ APIs across all Intel hardware

- Optimizes data ingestion & algorithmic compute together for highest performance
- Supports offline, streaming & distributed usage models for a range of application needs
- Flexible interfaces to leading big data platforms including Spark*
- Split analytics workloads between edge devices & cloud to optimize overall application throughput

High Performance Machine Learning & Data Analytics Library

Open Source, Apache* 2.0 License
MACHINE LEARNING ALGORITHMS

Regression
- Linear Regression
- Ridge Regression

Supervised learning
- Decision Tree
- Random Forest
- NEW - DAAL 2020 GradientBoosting

Classification
- Naïve Bayes
- Logistic Regression
- kNN
- SVM

Unsupervised learning
- Boosting (Ada, Brown, Logit)
- Naïve Bayes
- Logistic Regression
- kNN
- SVM

Collaborative filtering
- Alternating Least Squares
- Apriori

NEW - DAAL 2020
- LASSO
- ElasticNet

NEW - DAAL 2020u1
- GradientBoosting

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Open Source, Apache* 2.0 License
**DATA TRANSFORMATION & ANALYSIS ALGORITHMS**

**Basic statistics for datasets**
- Low order moments
- Quantiles
- Order statistics

**Correlation and dependence**
- Cosine distance
- Correlation distance
- Variance-Covariance matrix

**Matrix factorizations**
- SVD
- QR
- Cholesky

**Dimensionality reduction**
- PCA
  - NEW - DAAL 2020
  - tSVD

**Outlier detection**
- Univariate
- Multivariate

**Optimization solvers**
- (SGD, AdaGrad, lBFGS)

**Math functions**
- (exp, log,...)

*Algorithms supporting batch processing*

*Algorithms supporting batch, online and/or distributed processing*
Intel® DAAL 2019 Log Scale Optimization of Scikit-learn*

Intel® Data Analytics Acceleration Library 2019 (Intel® DAAL) Speedup vs XGBoost*

Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/intel-lab. Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/inside.

Intel® DAAL - PERFORMANCE

Optimization Notice

Correlation K-means Linear Regression Binary SVM Multi-Class SVM SVM

Log Speedup Factor (Optimized/Stock)

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>343</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® DAAL 2019 Log Scale Optimization of Scikit-learn*

Intel® Data Analytics Acceleration Library 2019 (Intel® DAAL) Speedup vs XGBoost*

Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/inside.

Intel® DAAL - PERFORMANCE

Optimization Notice

Correlation K-means Linear Regression Binary SVM Multi-Class SVM SVM

Log Speedup Factor (Optimized/Stock)

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>343</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® DAAL 2019 Log Scale Optimization of Scikit-learn*

Intel® Data Analytics Acceleration Library 2019 (Intel® DAAL) Speedup vs XGBoost*

Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/inside.

Intel® DAAL - PERFORMANCE

Optimization Notice

Correlation K-means Linear Regression Binary SVM Multi-Class SVM SVM

Log Speedup Factor (Optimized/Stock)

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>343</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® DAAL 2019 Log Scale Optimization of Scikit-learn*

Intel® Data Analytics Acceleration Library 2019 (Intel® DAAL) Speedup vs XGBoost*

Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/inside.

Intel® DAAL - PERFORMANCE

Optimization Notice

Correlation K-means Linear Regression Binary SVM Multi-Class SVM SVM

Log Speedup Factor (Optimized/Stock)

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>343</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® DAAL 2019 Log Scale Optimization of Scikit-learn*

Intel® Data Analytics Acceleration Library 2019 (Intel® DAAL) Speedup vs XGBoost*

Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/inside.

Intel® DAAL - PERFORMANCE

Optimization Notice

Correlation K-means Linear Regression Binary SVM Multi-Class SVM SVM

Log Speedup Factor (Optimized/Stock)

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>343</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® DAAL 2019 Log Scale Optimization of Scikit-learn*

Intel® Data Analytics Acceleration Library 2019 (Intel® DAAL) Speedup vs XGBoost*

Performance results are based on testing as of July 9, 2019 and may not reflect all publically available security updates. See configuration disclosures for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as TPC benchmarks, were measured using specific computer systems, components, software, operating systems, and functions. Any change in any of these factors may affect the results. Intel does not assure other information or performance results to assist you in fully evaluating your contemplated purchases, including the performance of that product in comparison with other products. Additional information at www.intel.com/inside.
FASTER, SCALABLE CODE WITH INTEL® MATH KERNEL LIBRARY

LINEAR ALGEBRA
- BLAS
- LAPACK
- ScALAPACK
- Sparse BLAS
- Iterative sparse solvers
- PARDISO*
- Cluster Sparse Solver

FFTS
- Multidimensional
- FFTW interfaces
- Cluster FFT

VECTOR RINGs
- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

SUMMARY STATISTICS
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

VECTOR MATH
- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

AND MORE
- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver
INTEL® MATH KERNEL LIBRARY FOR DEEP NEURAL NETWORKS

For developers of deep learning frameworks featuring optimized performance on Intel hardware

Distribution Details

▪ Open Source
▪ Apache* 2.0 License
▪ Common DNN APIs across all Intel hardware.
▪ Rapid release cycles, iterated with the DL community, to best support industry framework integration.
▪ Highly vectorized & threaded for maximal performance, based on the popular Intel® Math Kernel Library.

github.com/01org/mkl-dnn

Examples:
- Direct 2D Convolution
- Local response normalization (LRN)
- Rectified linear unit neuron activation (ReLU)
- Maximum pooling
- Inner product

Accelerate Performance of Deep Learning Models
Popular DL Frameworks are now optimized for CPU!

See installation guides at ai.intel.com/framework-optimizations/

More under optimization: Caffe2, PyTorch, PaddlePaddle

SEE ALSO: Machine Learning Libraries for Python (Scikit-learn, Pandas, NumPy), R (Cart, randomForest, e1071), Distributed (MLib on Spark, Mahout)

*Limited availability today

Other names and brands may be claimed as the property of others.
INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

DEEP LEARNING

- Caffe
- TensorFlow
- ONNX
- mxnet
- KALDI

- Model Optimizer
- Inference Engine
- Supports 100+ public models, incl. 30+ pretrained models

COMPUTER VISION

- OpenCV
- OpenCL™
- OpenVX

- Computer vision library (kernel & graphic APIs)
- Optimized media encode/decode functions

SUPPORTS MAJOR AI FRAMEWORKS

- Rapid adoption by developers

CROSS-PLATFORM FLEXIBILITY

- Multiple products launched based on this toolkit

HIGH PERFORMANCE, HIGH EFFICIENCY

- Breadth of product portfolio

Strong Adoption + Rapidly Expanding Capability

software.intel.com/openvino-toolkit
Obtain open source version at 01.org/openvinotoolkit

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT
For Deep Learning Inference – Part of Intel® Distribution of OpenVINO toolkit

**Model Optimizer**
- **What it is:** A Python*-based tool to import trained models and convert them to Intermediate representation.
- **Why important:** Optimizes for performance/space with conservative topology transformations; biggest boost is from conversion to data types matching hardware.

**Inference Engine**
- **What it is:** High-level inference API
- **Why important:** Interface is implemented as dynamically loaded plugins for each hardware type. Delivers best performance for each type without requiring users to implement and maintain multiple code pathways.

---

**Trained Models**
- Caffe*
- TensorFlow*
- MxNet*
- ONNX* (Pytorch, Caffe2 & more)
- Kaldi*

**Model Optimizer**
Convert & Optimize

Model Optimizer
IR
IR = Intermediate Representation format
.data
Load, infer

---

**Inference Engine**
Common API (C++ / Python)
Optimized cross-platform inference

- CPU Plugin
- GPU Plugin
- FPGA Plugin
- NCS Plugin
- GNA Plugin
- VAD Plugin

---

Extendibility C++
Extendibility OpenCL™
Extendibility OpenCL™

GPU = Intel CPU with integrated graphics processing unit/Intel® Processor Graphics

---

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
**PERFORMANCE OF PYTHON**

Python

Interpreter
GIL (constraining parallelism)

50x-5000x performance gap (or even more)

C

Optimizing compiler
OpenMP*/TBB/pthreads
PARALLELISM MATTERS MOST

Chapter 19: Performance Optimization of Black—Scholes Pricing

Unlocking parallelism is essential to make Python useful in production

Configuration info: - Versions: Intel® Distribution for Python 2.7.10 Technical Preview 1 (Aug 03, 2015), icc 15.0; Hardware: Intel® Xeon® CPU E5-2698 v3 @ 2.30GHz (2 sockets, 16 cores each, HT=OFF), 64 GB of RAM, 8 DIMMS of 8GB@2133MHz; Operating System: Ubuntu 14.04 LTS.

\[

\begin{align*}

C_{\text{all}} &= S_0 \cdot \text{CDF}(d_1) - e^{-r \cdot T} \cdot X \cdot \text{CDF}(d_2) \\
C_{\text{par}} &= e^{-r \cdot T} \cdot X \cdot \text{CDF}(-d_2) - S_0 \cdot \text{CDF}(-d_1)
\end{align*}

\]

\[

\begin{align*}

d_1 &= \frac{\ln\left(\frac{S_0}{X}\right) + \left(r + \sigma^2/2\right)T}{\sigma \sqrt{T}} \\
D_2 &= \frac{\ln\left(\frac{S_0}{X}\right) + \left(r - \sigma^2/2\right)T}{\sigma \sqrt{T}}
\end{align*}

\]
TWO INGREDIENTS TO GET CLOSE-TO-NATIVE PERFORMANCE

- **Pure Python**
  - Serial
  - Interpreted

- **Python + Libraries**
  - Partially Ninja-level
  - Partially Interpreted

- **Libraries + JITC**
  - Largely Ninja-level
  - 100% native

- **C++**
  - 100% Ninja-level
  - 100% native

*Optimization Notice*
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.*
PERFORMANCE OF PYTHON

Python + Numba*
http://numba.pydata.org/

LLVM-based compiler
Multiple threading runtimes

Small %% performance gap

C

Optimizing compiler
OpenMP*/TBB/pthreads

9 @numba.jit(nopython=True, parallel=True)
10 def logistic_regression(Y, X, w0, step, iterations):
11     """SGD solver for binary logistic regression."""
12     w = w0.copy()
13     for i in range(iterations):
14         w += step * np.dot((1.0/(1.0 + np.exp(Y * np.dot(X, w)))) * Y, X)
15     return w

High Performance Python

Python Libraries

Intel® Performance Libraries

Thin layer in Python or Cython

Native highly optimized libraries (Intel® MKL, Intel® DAAL, Intel® IPP)

More nodes, more cores, more threads, wider vectors, ...
CLOSE TO NATIVE CODE UMATh PERFORMANCE WITH INTEL PYTHON 2019

Compared to Stock Python Packages on Intel® Xeon Processors

Optimization Notice

Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Configuration: Stock Python: python 3.6.6 hc3d631a_0 installed from conda, numpy 1.15, numba 0.39.0, llvmlite 0.24.0, scipy 1.1.0, scikit-learn 0.19.2 installed from pip; Intel Python: Intel Distribution for Python 2019 Gold: python 3.6.5 intel_11, numpy 1.14.3 intel_py36_5, mkl 2019.0 intel_101, mkl_fft 1.0.2 intel_np114py36_6, mkl_random 1.0.1 intel_np114py36_6, numba 0.39.0 intel_np114py36_6, llvmlite 0.24.0 intel_py36_0, scipy 1.1.0 intel_np114py36_6, scikit-learn 0.19.1 intel_np114py36_35; OS: CentOS Linux 7.3.1611, kernel 3.10.0-514.el7.x86_64; Hardware: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (2 sockets, 18 cores/socket, HT:off), 256 GB of DDR4 RAM, 16 DIMMs of 16 GB@2666MHz

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Source: Intel Corporation - performance measured in Intel labs by Intel employees. Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804.
Intel® Distribution for Python*
https://software.intel.com/en-us/distribution-for-python

Accelerated NumPy, SciPy
Intel® MKL
Intel® C and Fortran compilers
Linear algebra, universal functions, FFT

Accelerated Scikit-Learn
Intel® MKL
Intel® C and Fortran compilers
Intel® Data Analytics Acceleration Library (DAAL)
via NumPy/Scipy

Solutions for efficient parallelism
TBB4py
github.com/IntelPython/smp
Intel® MPI library

Python APIs for Intel® MKL functions
github.com/IntelPython/mkl_fft
github.com/IntelPython/mkl_random
github.com/IntelPython/mkl-service [*]

Python APIs for Intel® DAAL
github.com/IntelPython/daal4py

Numba with upstreamed Intel contributions
Parallel Accelerator
support for SVML
support for TBB/OpenMP threading runtimes

conda create –c intel intelpython3_full
docker pull intelpython/intelpython3_full
pip install intel-numpy intel-scipy intel-scikit-learn


Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
HANDS-ON PREPARATION
Hands-On Sessions are for You!

Take your time to understand the Python code samples – don’t just execute Jupyter cells 1 by 1.

Also... there are solution files available, while it is in your own interest trying to find a solution yourself ...
INSTALL ON YOUR LOCAL MACHINE

- For installing the software on your own system:

  1. You need to have Conda on your system:


- Install Intel optimized software

  conda create -n intel-py -c intel intelpython3_core==2019.4
ACCESS THE JUPYTER SERVER

- Open your browser and go to the following url: http://IP_ADDRESS

- Choose a Username and Password for your access
- Then Sign in
• Go to the tab New for creating a Terminal window
ACCESS THE JUPYTER SERVER

• Go to the tab New for creating a Terminal window
• Type the following in the Terminal Window to copy the samples:

```
 cp -r /srv/intel-ai-workshop/* .
```
Numpy-Numba: black-scholes.ipynb

1) Why is the performance using the NumPy functions is lower than expected?
2) Implement the black_scholes function in a NumPy like fashion
3) Measure the speedup and explain where exactly it is coming from
4) Implement the black_scholes function using Numba
DATA ANALYSIS AND MACHINE LEARNING

Data Input
- Pandas
- Spark
- HPAT

Data Preprocessing

Model Creation
- SciKit-Learn
- Spark
- DL-frameworks
- daal4py

Prediction

*Other names and brands may be claimed as the property of others.

more nodes, more cores, more threads, wider vectors, ...
THE MOST POPULAR ML PACKAGE FOR PYTHON*

scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
Identifying to which category an object belongs to.
Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ...

Regression
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ...

Clustering
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes.
Algorithms: k-Means, spectral clustering, mean-shift, ...

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
DAAL4PY: ACCELERATED ANALYTICS TOOLS FOR DATA SCIENTISTS

• Package created to address the needs of Data Scientists and Framework Designers to harness the Intel® Data Analytics Acceleration Library (DAAL) with a Pythonic API

• Pandas compatible, one-liner API for accessing many hardware accelerated Machine Learning and Analytics functions

• Powers our Scikit-Learn* accelerations in our shipped version of the package

• Extends capabilities past Scikit-learn by providing scaling and distributed modes
ACCELERATING MACHINE LEARNING

- Efficient memory layout via Numeric Tables
- Blocking for optimal cache performance
- Computation mapped to most efficient matrix operations (in MKL)
- Parallelization via TBB
- Vectorization

Try it out! conda install -c intel scikit-learn
ACCELERATING K-MEANS

Performance speedups for Intel® Distribution for Python* scikit-learn on Google Cloud Platform's 96 vCPU instance Intel® Xeon™ Processors

- PCA-based: 23X faster
- random: 21X faster
- k-means++: 22X faster

System Configuration: GCP VM, zone us-central1-c, 96 vCPU, Intel Skylake; 360 GB memory. Ubuntu 16.04.3 LTS, Linux instance-1 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux Intel® Distribution for Python* from Docker Image intelpython/intelpython3_fulltest (created 2017-09-12T20:10:42.8629655592Z); Stock Python*: pip install scikit-learn

DAAL4PY

Fast & Scalable
- Close to native performance through Intel® DAAL
- Efficient MPI scale-out
- Streaming

Easy to use
- Known usage model
- Picklable

Flexible
- Object model separating concerns
- Plugs into scikit-learn
- Plugs into HPAT

Open
- Open source: https://github.com/IntelPython/daal4py

https://intelpython.github.io/daal4py/
ACCELERATING SCIKIT-LEARN THROUGH DAAL4PY

Scikit-Learn Equivalents

Scikit-Learn API Compatible

daal4py

Intel® DAAL

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

> python -m daal4py <your-scikit-learn-script>

Monkey-patch any scikit-learn on the command-line

import daal4py.sklearn
daal4py.sklearn.patch_sklearn()

Monkey-patch any scikit-learn programmatically

https://intelpython.github.io/daal4py/
SCALING MACHINE LEARNING BEYOND A SINGLE NODE

Simple Python API
Powers scikit-learn
Powered by DAAL
Scalable to multiple nodes

Try it out! conda install -c intel daal4py
MACHINE LEARNING ALGORITHM: K-MEANS
Machine Learning Workflow

Build your data set
- Query a database
- Choose the data format
- Store the data in a file
- Clean the data
- ...

Features Extraction
- Explore the data
- Select the important features
- ...

Validation set

Training set

Test set

Training

Tuning the parameters

Evaluation the model

Machine Learning

Model

Splitting the data

Inference and Deployment

New data collection

Features Extraction

Prediction

Target

Predicted data

INFE RENCE AND DE PLOYMENT
K-MEANS ALGORITHM

• K-means is a clustering method
• It is used to group objects in a (high-dimensional) sample
• K-means is based on centroids

Interesting use cases for k-means

• Document classification
  • Similarity identification
• Customer segmentation
  • Identifying patterns of interest areas
• Insurance fraud detection
  • Isolate new claims based on proximity to past fraud

source: https://dzone.com/articles/10-interesting-use-cases-for-the-k-means-algorithm
K-MEANS ALGORITHM DESCRIPTION

1. k (number of clusters in the dataset) is an external free parameter

2. k random objects are associated with initial centroids

3. each object of the dataset is assigned to form a cluster with its closest centroid

4. new centroids are computed by taking the average position of the objects in a given cluster

5. Evaluate convergence condition

6. Output: centroids location and association of the objects

Pros and cons:

• Simple and relatively robust

• Finding the best k is not trivial

• Results might depend on initial centroids

• Sensitive to outliers and to features with different dynamical range → data preparation
K-MEANS EXAMPLE: COLOR QUANTIZATION

• An interesting application of K-means is to reduce the number of colors in a figure, while preserving the overall quality

• Initial data: every pixel has three color channels (RGB) expressed with 8 bit (0...255)

• For comparison, a clustering based on randomly picked colors will be shown

The image of the Summer Palace (China) is among the sample data contained on SciKit-Learn
HANDS-ON
**K-MEANS EXAMPLE: COLOR QUANTIZATION**

kmeans.ipynb

The hands-on is based on six main steps:

1) Data preparation  
2) Computing the cluster centers  
3) Labelling the data  
4) From scikit-learn to daal4py: command line  
5) From scikit-learn to daal4py: monkey-patch in the script  
6) K-means with daal4py: kmeans-daal4py.py
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Copyright © 2019, Intel Corporation. All rights reserved. Intel, the Intel logo, Pentium, Xeon, Core, VTune, OpenVINO, Cilk, are trademarks of Intel Corporation or its subsidiaries in the U.S. and other countries.
BACKUP